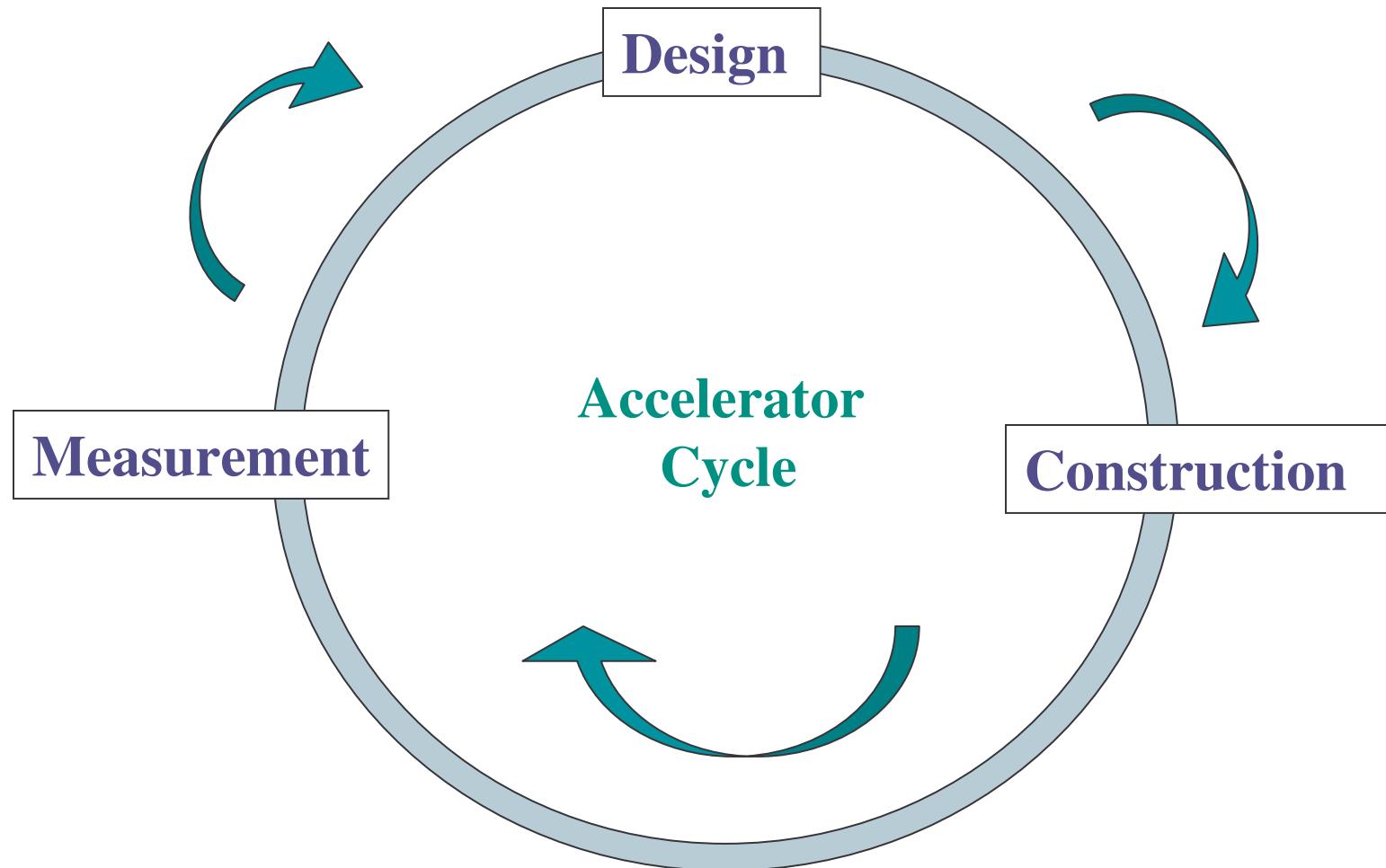


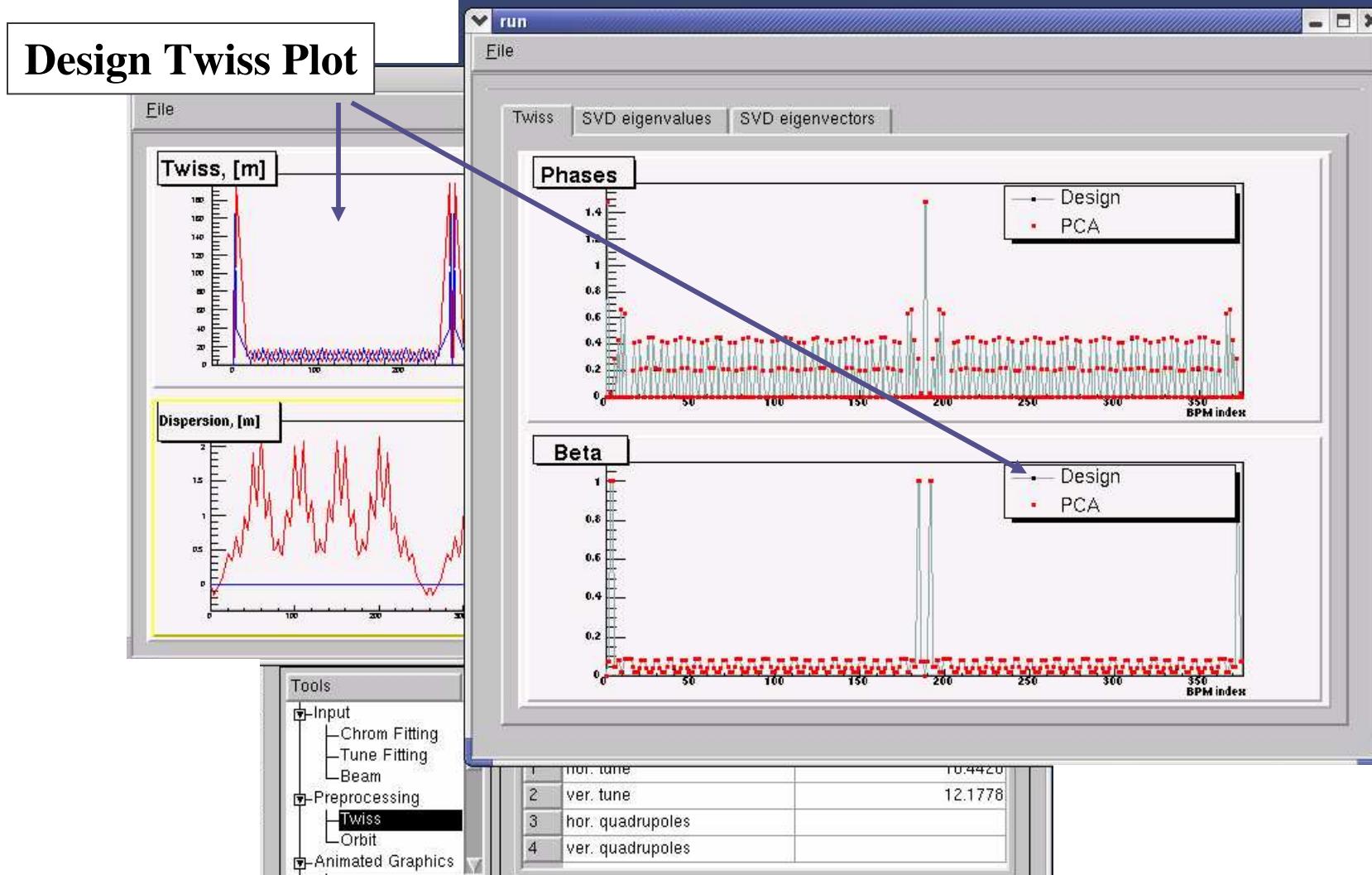
Section 3: Instrumental Analysis of 1D Particle and Bunch Motion

- Objectives
- APDF-based framework
 - teapot.apdf
 - Adding and editing the Noisy Monitor tracker
- Simulation
 - Demo
 - Examples

Objectives



Model-Independent vs Model-based Analysis



2. Accelerator Propagator Description Format (APDF)

teapot.apdf

```
<apdf>
<propagator id="teapot" accelerator="ring">
  <create>
    <link algorithm="TEAPOT::DriftTracker" types="Default" />
    <link algorithm="TEAPOT::DriftTracker" types="Marker|Drift" />
    <link algorithm="TEAPOT::DipoleTracker" types="Sbend" />
    <link algorithm="TEAPOT::MltTracker" types="Quadrupole|Sextupole"/>
    <link algorithm="AIM::Monitor" types="[VH]monitor" />
    <link algorithm="AIM::PoincareMonitor" types="Monitor" />
  </create>
</propagator>
</apdf>
```

trackers

element types

To start this application:

```
./linux/run ring ../../lattices/collider.adxf ./apdf/teapot.apdf
```

2.1 Interface of the Noisy Monitor class

[./src/NoisyMonitor.hh](#)

```
class NoisyMonitor : public AIM::Monitor {  
  
public:  
    ...  
  
    /** Propagates a bunch of particles */  
    void propagate (UAL::Probe& probe);  
  
protected:  
    int m_seed;  
    double ran1(int& iseed);  
};
```

Tracking method

```
class NoisyMonitorRegister  
{  
public:  
    NoisyMonitorRegister();  
};
```

The secondary class used for the registration of the NoisyMonitor class in the collection of propagators.

2.2 The propagate method of the Noisy Monitor class

~~./src/NoisyMonitor.cc~~

```
void UAL::USPAS::NoisyMonitor::propagate(UAL::Probe& probe)
{
    PAC::Bunch& bunch = static_cast<PAC::Bunch&>(probe);

    int counter = 0;
    PAC::Position pos; ←————— Container with 6 coordinates
    for(int i=0; i < bunch.size(); i++){
        if(bunch[i].isLost()) continue;
        pos += bunch[i].getPosition();
        counter++;
    }

    pos /= counter;
    double x = pos.getX();

    pos.setX(x + 1.0*x*ran1(m_seed)); Adding a noise source

    m_data.push_back(pos);
}
```

2.3 APDF file with Noisy Monitor

[./apdf/teapot+noise.apdf](#)

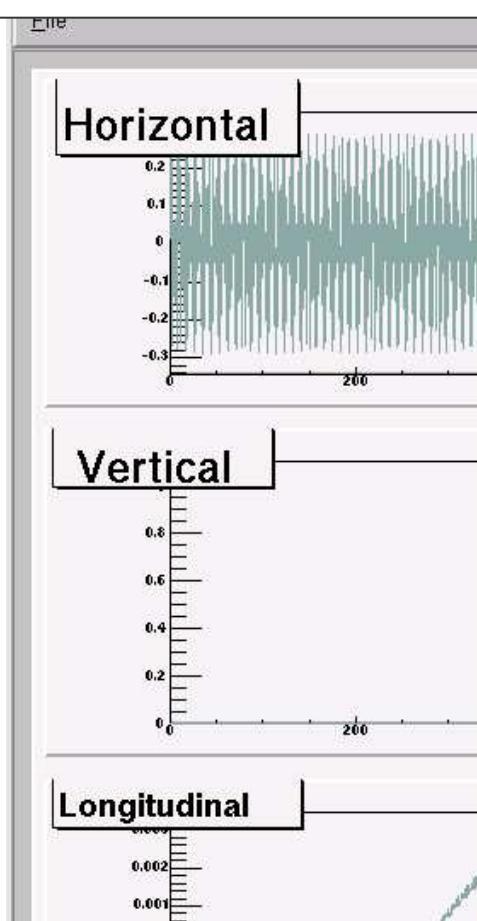
```
<apdf>
<propagator id="teapot_noise" accelerator="ring">
  <create>
    <link algorithm="TEAPOT::DriftTracker" types="Default" />
    <link algorithm="TEAPOT::DriftTracker" types="Marker|Drift" />
    <link algorithm="TEAPOT::DipoleTracker" types="Sbend" />
    <link algorithm="TEAPOT::MltTracker" types="Quadrupole|Sextupole|"/>
    <link algorithm="UAL::USPAS::NoisyMonitor" types="[VH]monitor" />
    <link algorithm="AIM::PoincareMonitor" types="Monitor" />
  </create>
</propagator>
</apdf>
```

To start this application:

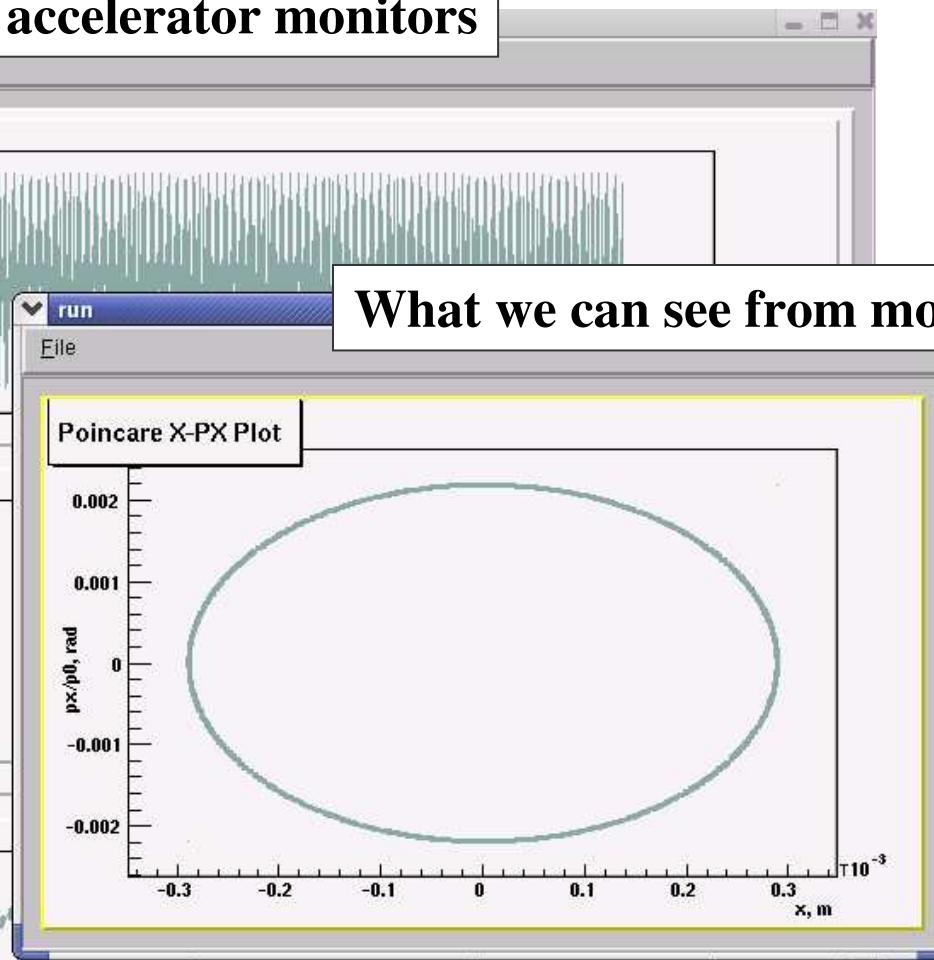
`./linux/run ring ..\lattices\collider.adxf ./apdf/teapot+noise.apdf`

3.1 Monitor turn-by-turn data vs Poincare Plot

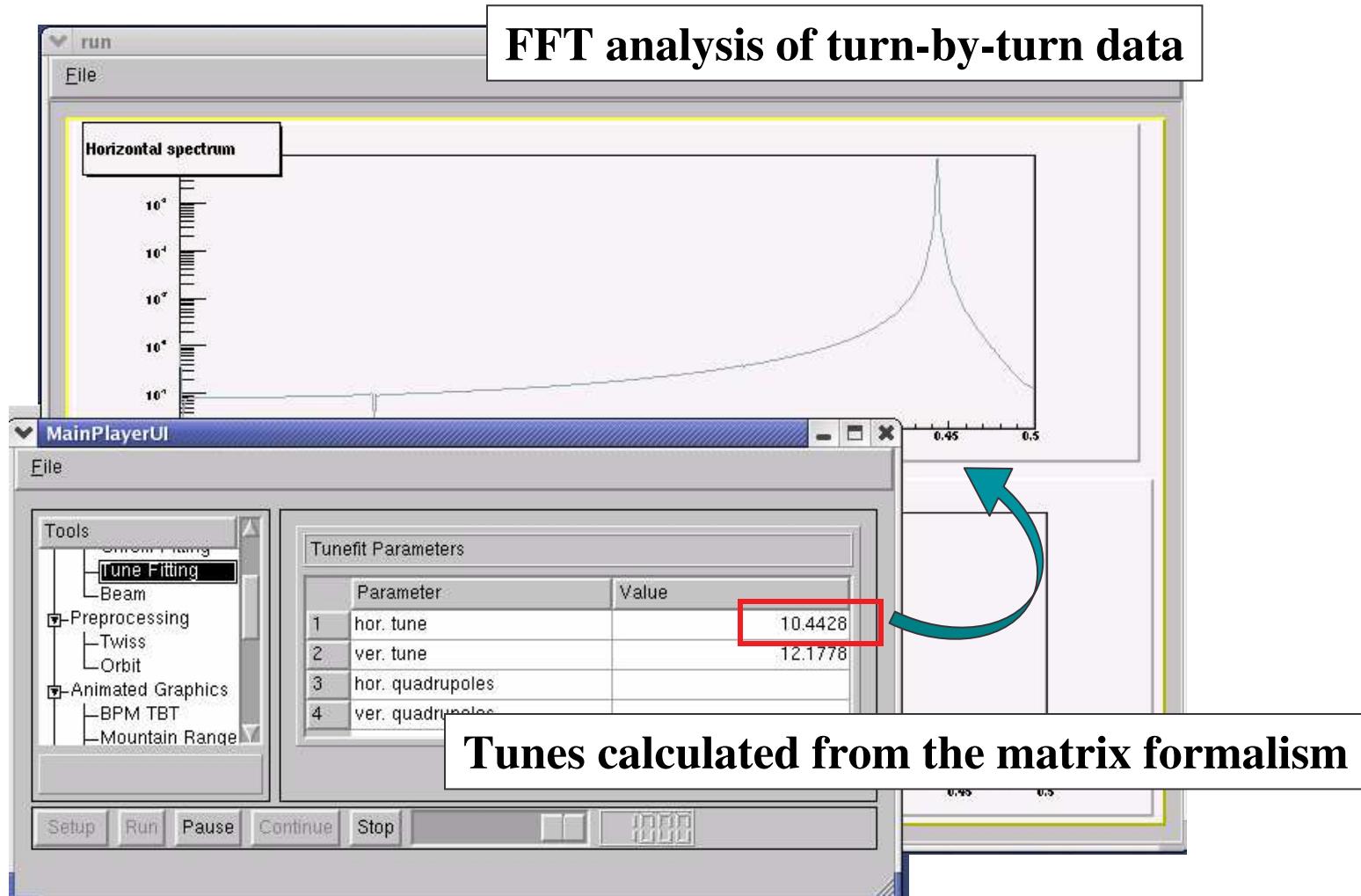
What we can see from accelerator monitors



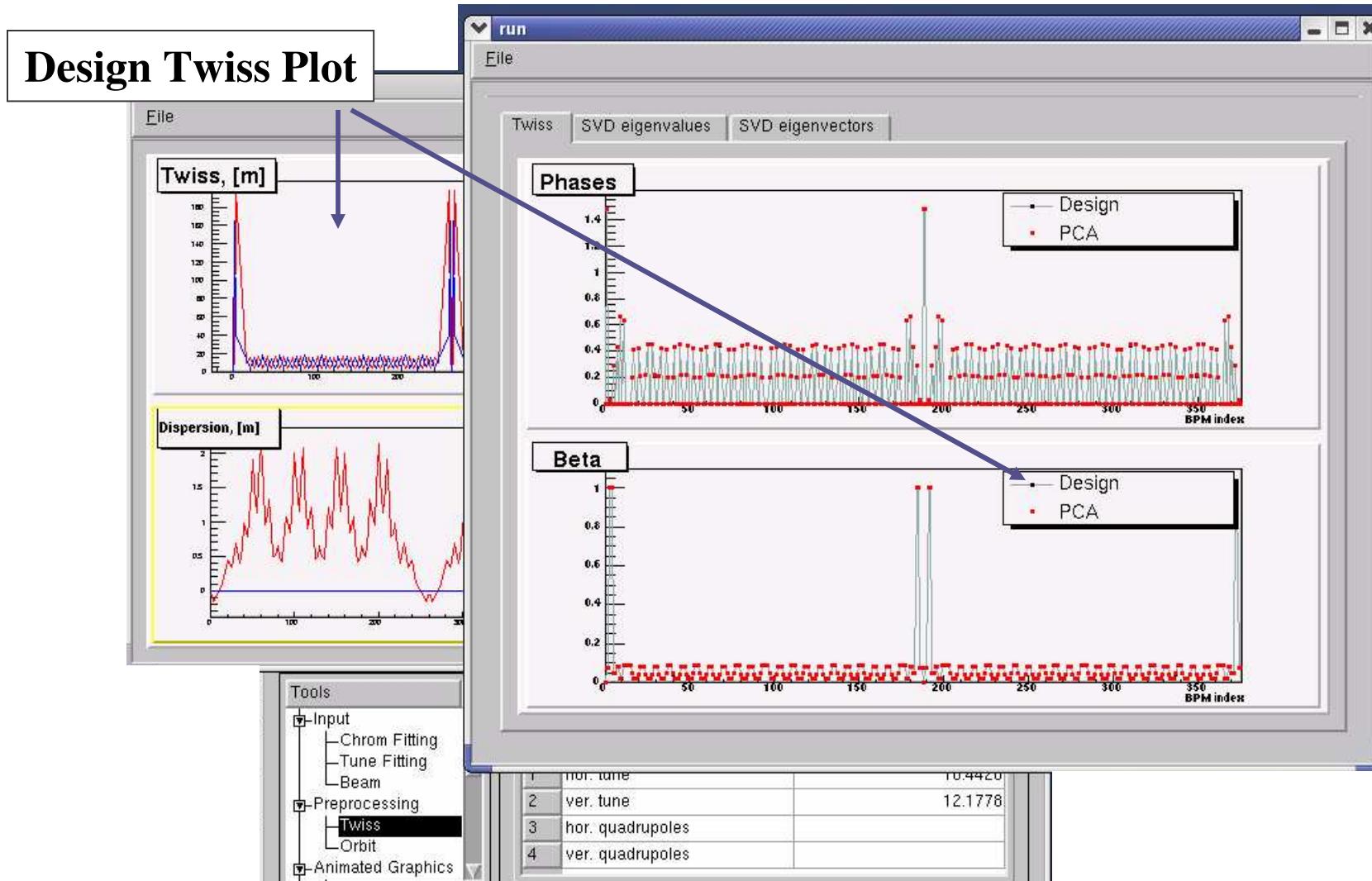
What we can see from model tracking



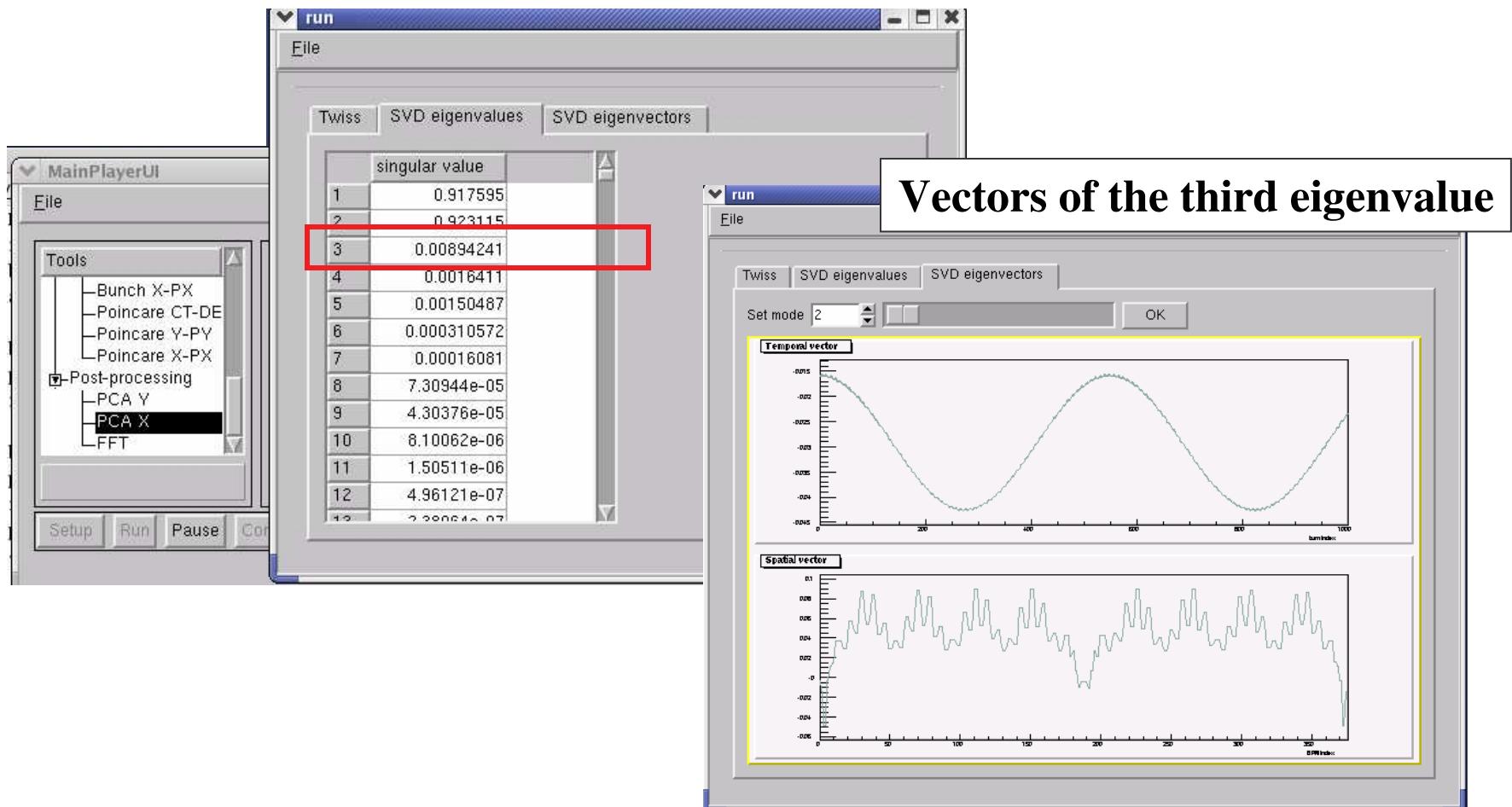
3.2. FFT vs TuneFit Matrix-based Tunes



3.3. PCA vs Design Twiss Functions



3.4 SVD eigenvalues, temporal and spatial vectors



Simulation Examples

- **S 3.1:** Run the instrumental analysis application with the teapot.apdf file and investigate associated plots, such as Turn-by-turn, Poincare, Twiss, FFT, and PCA.
- **S 3.2:** Zero a RF voltage in the run.cc file, recompile application, repeat S 3.1 and compare results.
- **S 3.3:** Vary the noise level in the Noisy Monitor, run the application with the teapot+noise.apdf file, and investigate FFT and PCA results.
- **S 3.4:** Change the damping rate in the Damping Tracker and repeat the S 3.1 simulation with the teapot+damping.apdf file